

The Influence of Size and Coverage on Test Suite Effectiveness

Akbar Siami Namin
Department of Computer Science
Texas Tech University at Abilene
302 Pine St., Abilene, TX 79601
akbar.namin@ttu.edu

James H. Andrews
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 2T2
andrews@csd.uwo.ca

ABSTRACT

We study the relationship between three properties of test suites: size, structural coverage, and fault-finding effectiveness. In particular, we study the question of whether achieving high coverage leads directly to greater effectiveness, or only indirectly through forcing a test suite to be larger. Our experiments indicate that coverage is sometimes correlated with effectiveness when size is controlled for, and that using both size and coverage yields a more accurate prediction of effectiveness than size alone. This in turn suggests that both size and coverage are important to test suite effectiveness. Our experiments also indicate that no linear relationship exists among the three variables of size, coverage and effectiveness, but that a nonlinear relationship does exist.

Categories and Subject Descriptors

D.2.5.m [Software Engineering]: Testing and Debugging—
Test coverage of code

General Terms

Experimentation, Measurement

Keywords

Coverage Criteria, Statistical Analysis

1. INTRODUCTION

Structural coverage measures, such as block coverage and decision coverage, have long been used as adequacy criteria to measure the thoroughness of test suites. The use of coverage measures as adequacy criteria is based on the belief that a test suite that causes more elements of a program to be executed is likely to be more thorough.

In the last 20 years, this belief has been supported by a number of experiments that appeared to show a high correlation between coverage and test suite effectiveness [6, 9, 5,

2]. Here, effectiveness is usually being measured by the number of faulty versions of subject programs that can be detected by the test suite. The experiments indicated that test suites executing a higher percentage of program elements were more effective, and that test suites achieving 100% of feasible coverage on stronger coverage measures (such as def-use coverage) were more effective than those achieving 100% of feasible coverage on weaker coverage measures (such as block coverage).

However, correlation is not causation, and there is another factor that could potentially explain the effectiveness of high-coverage test suites: test suite size, as measured in number of test cases.

Adding a test case to a test suite makes it at least as effective as it was before, and possibly more effective. Therefore, on average, if test suite A is bigger (contains more test cases) than test suite B, A will be at least as effective as B. However, adding a test case to a test suite will also make it achieve at least as high coverage, and possibly more; and in order to achieve higher coverage, it is often necessary to increase the number of test cases in a test suite. A similar statement can be made about strength of coverage measures: stronger coverage measures often require adding test cases not needed by weaker coverage measures. Therefore, simply showing a correlation between coverage and effectiveness is not sufficient to motivate the use of coverage as an adequacy criterion. Coverage could instead merely cause increased size, which in turn is the direct cause of increased effectiveness; or the third factor of size could be causing both higher coverage and higher effectiveness.

In the work reported on in this paper, we attempted to determine how size and coverage separately influence test suite effectiveness, by performing experiments on subject software. We generated test suites of fixed size but varying coverage, and measured the correlation of coverage and effectiveness when holding size constant. This experiment indicated that for many subject programs and coverage measures, there was a moderate to very high correlation between coverage and effectiveness, usually at low test suite sizes. However, the experiment unexpectedly also showed that there were major differences in behaviour among the subject programs studied, including some subject programs for which there was always a low correlation between coverage and effectiveness for all coverage measures.

In order to obtain further insight into the question, we performed various complementary statistical analyses. These consistently showed that both size and coverage independently influenced effectiveness. We then fit various regres-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSTA '09, July 19–23, 2009, Chicago, Illinois, USA.
Copyright 2009 ACM 978-1-60558-338-9/09/07 ...\$10.00.

sion models to the data thus collected, in order to see whether models taking account of both size and coverage were better fits than those taking account only of size, or only of coverage. We found that the best model was one that suggested a particular nonlinear relationship between the three variables in question.

As case studies, we then applied the resulting analysis to two other subject programs, in order to determine whether the results extended to them. We did so by again fitting various linear regression models to the data and comparing them. We found the same nonlinear relationship between size, coverage and effectiveness that we found with the earlier programs.

A summary of our primary observations are as follows:

1. We observed that both size and coverage independently exert an influence on test suite effectiveness.
2. We observed that a correlation between coverage and effectiveness is dependent on the software under study and on the range of test suite sizes used.
3. We did not observe a linear relationship amongst the three variables of size, coverage and effectiveness.
4. However, we did observe a linear relationship amongst $\log(\text{size})$, coverage and effectiveness; this relationship held across programs in both C and C++, and across various ways of measuring coverage and effectiveness.

A discussion of previous work touching on these questions is contained in Section 2. Section 3 describes the setup of our main experiment. Section 4 describes the statistical analysis we performed on the data resulting from the experiment. Section 5 describes the context and results of our two case studies. Section 6 discusses the implications of our work and the threats to validity of the results. Section 7 concludes.

2. RELATED WORK

Early papers that supported a relationship between coverage and effectiveness were those by Frankl and Weiss [6], Hutchins et al. [9], and Frankl and Iakounenko [5]. These papers measured the effectiveness of a test suite as the fraction of faulty versions of subject programs that were revealed by the test suite; in this paper, we refer to this number as the *fault adequacy ratio*, or AF. These papers showed that test suites that achieved higher coverage tended to achieve a higher AF. Two of the papers [6, 5] control for size by choosing test suite sizes or size ranges, and graphing coverage against effectiveness. The graphs suggest that effectiveness is usually constant until high coverage levels are achieved, at which point it increases rapidly. Frankl and Weiss [6] also generate non-linear models from their data using logistic regression, but there is no clear pattern to the resulting regression models.

Andrews et al. [1] showed that automatically-generated mutants of programs [11] can act similarly to real faults. They argued that using mutants to measure adequacy makes experiments more reproducible, and since it leads to a larger number of faulty versions, also makes it easier to find statistically significant properties of test suites. They then [2] studied the relationship between coverage and effectiveness, using the *mutant adequacy ratio*, AM, as a measure of effectiveness instead of AF. They confirmed the earlier linkage between coverage and effectiveness.

Table 1: Description of subject programs. NLOC: Net lines of code. NTC: Number of test cases. NMG: Number of mutants generated by all mutation operators. NM: Number of selected mutants that were non-equivalent.

Program	NLOC	NTC	NMG	NM
<code>printtokens</code>	343	4130	11741	1551
<code>printtokens2</code>	355	4115	10266	1942
<code>replace</code>	513	5542	23847	1969
<code>schedule</code>	296	2650	4130	1760
<code>schedule2</code>	263	2710	6552	1497
<code>tcas</code>	137	1608	4935	4935
<code>totinfo</code>	281	1052	8767	1740

Andrews et al. [2] also noted that test suites constructed by picking a series of test cases that each increase coverage tended to be more effective than test suites of the same size constructed by randomly picking test cases. However, this result lends at best indirect support to the claim that coverage is a good adequacy criterion, since the coverage-based test suites were constructed using a particular algorithm. It might be that the algorithm, rather than the mere fact of high coverage, was what caused the greater effectiveness.

Earlier research by Rothermel et al. [13] gives the same indirect support. The researchers studied test suites that were reduced in size by eliminating test cases while preserving coverage. They found that these test suites were more effective than those that were reduced to the same size by randomly eliminating test cases. The support is again indirect because the increase in effectiveness might be a result of the method of construction.

3. EXPERIMENTAL PROCEDURE

In this section, we describe the subject software of our main experiment, and how we collected our data.

3.1 Subject Software

The subject software for our first experiment was the well-known Siemens suite of seven C programs ranging in size from 137 to 513 net lines of code (lines of code without comments and whitespace). The subject software and most of the test cases associated with them were developed by Siemens researchers [9], and then test cases were added by Rothermel et al. [12] until each decision was covered by at least 30 test cases.

We accessed the Siemens programs and their associated test cases through the Subject Infrastructure Repository (SIR) at the University of Nebraska-Lincoln¹. See Table 1 for a description of these subject programs.

3.2 Data Collection

The ideal experiment to measure the separate influences of size and coverage on effectiveness would be to randomly generate test suites with each pair of (low size, high size) and (low coverage, high coverage), measure the effectiveness of each, and then perform a two-dimensional analysis of variance (ANOVA) to see if size, coverage or both influence effectiveness. Unfortunately, our initial investigations

¹Thanks to Hyunsook Do and Gregg Rothermel at the University of Nebraska for their help in accessing this repository.

revealed that this ideal experiment is not feasible: test suites with high coverage and low size, or low coverage and high size, have a vanishingly small probability of occurring naturally.

We therefore began by generating, for each subject program, 100 test suites of each size from 1 to 50 test cases, in which each test case was chosen randomly from the test pool. We chose 50 as the upper bound for the experiment because we observed that test suites of this size usually achieved very high coverage for our subject programs. We chose to perform random selection in order to remove any bias that may have been introduced by carrying out a specific test suite construction algorithm.

For each test suite, we measured the block, decision, C-use and P-use coverage of the test suite on the gold version of the program, using the tool ATAC [8]. Block coverage is the percentage of basic blocks that are executed by the test cases in the test suite; decision coverage is the percentage of decision directions (true or false) that are followed by test cases in the test suite. C-use and P-use coverage are dataflow coverage measures. A C-use pair is a statement assigning a value to a variable and a later statement that uses that value in a computation; a P-use pair is the same, except that the later statement uses the value in a predicate (decision). C-use and P-use coverage measure the percentage of C-use and P-use pairs actually executed. Table 2 shows the proportions of feasible coverage for all these measures on the seven Siemens programs.

For each subject program, we reused mutants from an earlier study [14] that we had generated using the Proteum mutant generator [4]. Table 1 also shows the number of mutants generated by Proteum. In our earlier study, we had started by using all mutants of **tcas**, and then randomly selected 2000 mutants of all other subject programs. The last column of Table 1 shows the number of mutants that we randomly selected that turned out to be non-equivalent to the gold version of the program. Since determining equivalence is in general undecidable, for simplicity, mutants that were not killed by any test case in the entire test pool were considered equivalent.

We ran all mutants on all test cases, and recorded which test cases detected which mutants, in the sense of forcing the mutant to give different output from the gold version. For each of the test suites we generated, we computed the mutation adequacy ratio AM as the fraction of these selected mutants that were detected by the test suite. This AM was the number that we used as a measurement of test suite effectiveness.

4. DATA ANALYSIS

We analyzed the resulting data primarily with the R statistical package [15]. We first performed various visualizations in order to get an overall sense of the data. We then performed an analysis of covariance (ANCOVA) in order to see whether both size and coverage influenced effectiveness, and a principal component analysis in order to see whether we could identify linear relationships among the variables.

Next we plotted the correlation between coverage and effectiveness when holding size constant, in order to detect any consistent patterns that might exist. Finally, we performed various linear regressions, on both the raw data and transformed data, in order to gain more insight into the precise relationships among the variables.

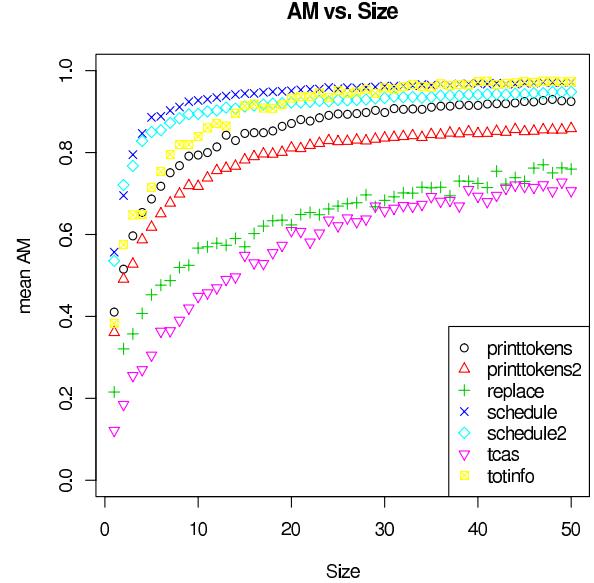


Figure 1: The relationship between size and effectiveness.

4.1 Visualizations

Figures 1 and 2 show how test suite size influences both effectiveness and coverage, for all seven subject programs. In Figure 1, each point represents the average AM for all 100 test suites of the given size; in Figure 2, each point represents the average percentage of feasible decision coverage for all 100 test suites of the given size (the graphs for the other coverage measures are similar).

In order to visualize the interrelationships among all three variables, we created three-dimensional scatter plots in which AM was on the vertical axis, and size and coverage were on the horizontal axes. We did this for each subject program and each coverage measure.

Figure 3 shows two typical such plots, for programs **replace** and **schedule** and the block coverage measure. The data points are evenly spaced on the Size axis because of our method of test suite generation, but as size increases, both coverage and effectiveness increase quickly. In the **replace** plot, note the presence of some test suites which achieve a very high effectiveness and high coverage, for all but the smallest sizes of test suites. We believe that these data points result from the presence of a small number of test cases that cover many lines of code and kill many mutants; these test cases are not often selected, but when they are, they increase the coverage and effectiveness of the test suite substantially. Three of the programs (**printtokens2**, **replace** and **tcas**) exhibited such test cases. The others exhibited a more continuous pattern of rising coverage and effectiveness.

4.2 ANCOVA

Because we had one continuous outcome variable (AM), one continuous factor (coverage), and one discrete factor (size), we performed analysis of covariance (ANCOVA) to see if either or both of the two factors influenced the outcome variable. Since ANCOVA builds a linear model, we must specify whether we want the linear model to include an intercept or not; no intercept indicates that we want the

Program	%Block	%Decision	%C-Uses	%P-Uses
printtokens	95(231/242)	94(102/108)	98(179/183)	93(146/157)
printtokens2	99(242/244)	98(158/161)	99(132/134)	99(207/210)
replace	97(277/287)	94(154/163)	94(367/389)	89(378/426)
schedule	99(159/161)	95(52/55)	99(114/115)	96(75/78)
schedule2	98(169/173)	94(83/88)	97(88/91)	94(83/88)
tcas	99(105/106)	90(45/50)	98(42/43)	91(31/34)
totinfo	96(145/151)	81(70/86)	93(141/152)	77(98/128)

Table 2: Proportions of feasible coverage for all criteria, as measured by ATAC. Each column is of the form $p(n/d)$, where p is the percentage of program elements covered, n is the number of program elements covered, and d is the total number of program elements identified.

Coverage Degree vs. Size – DEC

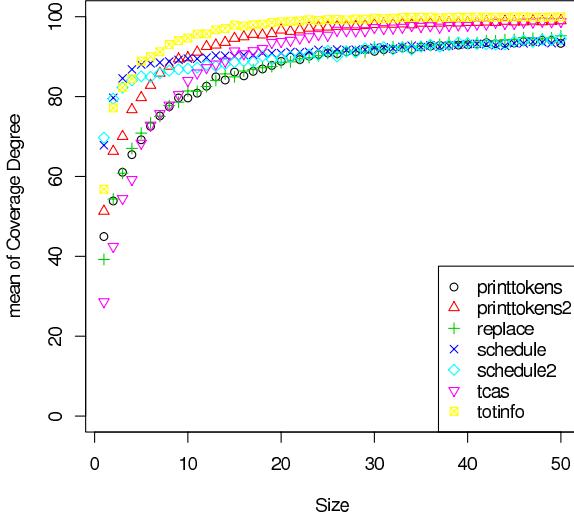


Figure 2: The relationship between coverage and size, for the decision coverage criterion.

fitted line to go through the origin. For each subject program and each coverage measure, we performed ANCOVA with no intercept, since at test suite size 0 (no test cases) we expect no coverage and an AM of zero².

ANCOVA yielded a value of p for the size factor, the coverage factor, and the interaction of size and coverage. A p value below 0.05 indicates that the factor or interaction does significantly influence the outcome variable. The p values in this case were always below 0.001 for the two factors of size and coverage, suggesting strongly that they always independently influenced effectiveness. The p values were always below 0.05 for the interaction of size and coverage except in three cases: block coverage of `schedule` and `tcas`, and decision coverage of `replace`. This indicated that there was often an interaction effect as well.

4.3 Principal Component Analysis

We performed a principal component analysis (PCA) on the data for each program. Although PCA is usually used

²We applied the `aov` function provided by the R system on the normalized data. As an example, the formula for block coverage is: `aov(AM ~ SIZE + BLK_Cov.Deg. + SIZE * BLK_Cov.Deg. - 1)`.

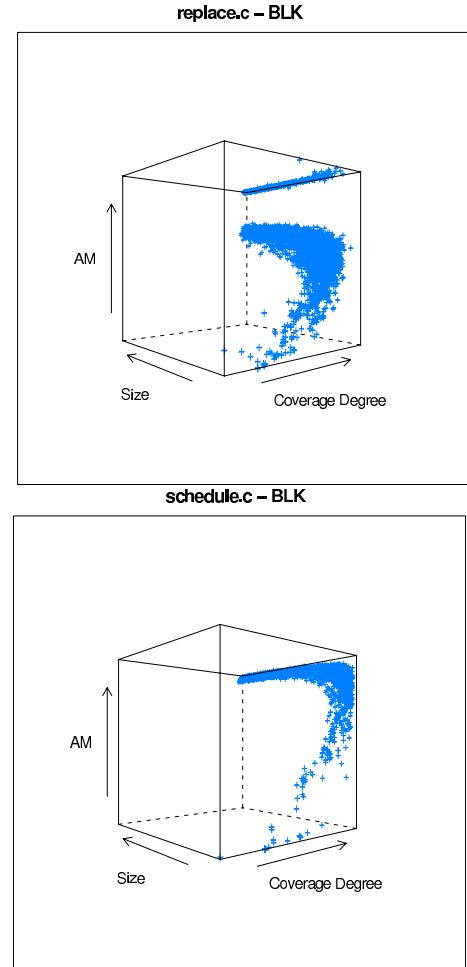


Figure 3: The relationship among the three variables for the block coverage criterion, for the subject programs `replace` (top) and `schedule` (bottom).

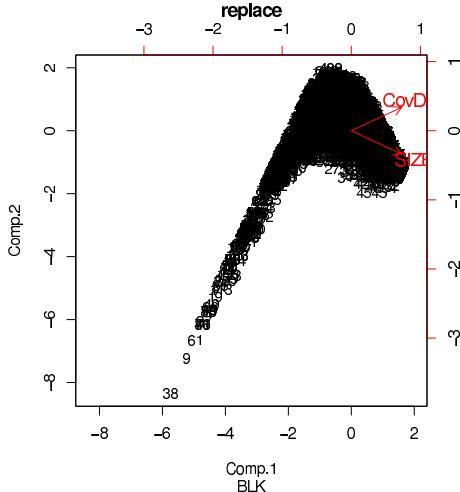


Figure 4: Biplot for principal component analysis of the data for the block coverage criterion, for the subject program `replace`.

for variable reduction, here we use it to see the contribution of each variable to the overall shape of the data points.

A typical resulting biplot is given in Figure 4, for the `replace` program and the block coverage measure. The curved “horseshoe” pattern of the data points confirms the intuition that we would get from the three-dimensional plots, that there is no strong linear relationship among the three variables of interest. However, the two arrows on the biplot show that both size and coverage contribute approximately equally to the first principal component, suggesting again that both influence effectiveness.

4.4 Correlation of Coverage and Effectiveness

Since the largest test suites for each subject program usually achieved close to the maximum coverage for the program, we suspected that the correlation between coverage and effectiveness for the large test suites would probably be low. However, we did not know whether this result would extend to small test suites.

We therefore calculated r^2 (the square of the Pearson correlation) and the Kendall rank correlation, between coverage and AM, for each size of test suite from 1 to 50, each coverage measure, and each subject program³. We then plotted these correlation values against test suite size.

Figure 5 shows these plots. There is one plot for each coverage measure, and each plot shows the data for all seven subject programs. Because we are interested here only in the overall trend, we show only the LOWESS (locally-weighted polynomial regression) smoothing curve approximating the data for each program, as fitted by the R system.

For five of the subject programs, the correlation values fell as the size of the test suite increased: for small test

³ r^2 measures how close two variables are to having a linear relationship, while the Kendall rank correlation measures how close the list of data points when ranked by one variable would be to the list ranked by the other variable. For both correlation measures, we use the standard Guilford scale [7] for verbal description; this scale describes correlation of under 0.4 as “low”, 0.4 to 0.7 as “moderate”, 0.7 to 0.9 as “high”, and over 0.9 as “very high”.

suites, the correlations between coverage and effectiveness were usually moderate to very high, and for large test suites, the correlations were low to moderate. However, for one of the subject programs (`schedule2`), the correlation value was always low, and for another of the subject programs (`schedule`), correlation sometimes fell and sometimes rose with increasing test suite size. The observations for Kendall rank correlation were similar. The reasons for this are not clear. `schedule` and `schedule2` were the two programs at the top of the AM vs. Size graph (Figure 1), indicating that they quickly achieve high coverage as more test cases are added. The uncharacteristic correlation behaviour may simply mean that most of the test suites we studied are very effective for these programs, and so there is no particular relationship between effectiveness and any other variable.

Such marked differences in behaviour amongst these well-studied subject programs were unexpected, and suggest that experimenters using different subject programs and ranges of test suite sizes are likely to reach different conclusions about the relationship between coverage and effectiveness.

4.5 Regression Models

A linear model of data is a linear equation of the form $A = B_1 \cdot C_1 + \dots + B_n \cdot C_n$, in which the C_i variables correspond to factors (input variables) and A corresponds to an outcome variable. *Linear regression* is the process of finding values for the coefficients B_i that best fit the actual data. In doing such a linear regression, we say that we are fitting a linear model to the data.

Using R’s linear regression feature, we fitted various linear models to the collected data, now using data for all test suite sizes together. We had two purposes in doing this. The first was to determine whether models taking account of coverage were better at predicting effectiveness than those not taking account of coverage. The second was to see whether transforming the data, for instance by the standard log transformation, would result in better models of effectiveness.

We first *normalized* the data by dividing all values of coverage by the standard deviation of coverage, and all values of size by the standard deviation of size. This allowed for a fair comparison between coverage and size, since the variance of size was larger and could otherwise have appeared to have an exaggerated influence.

We then generated linear regression models predicting each of AM and $\log(\text{AM})$, using combinations of size, $\log(\text{size})$, coverage, and $\log(\text{coverage})$. We measured how good these models were by observing the adjusted R^2 value reported by the R system. Adjusted R^2 indicates how well a model fits the actual data, while adjusting for the number of variables used in the model, in order to avoid bias in favour of models with more variables.

We found that models of the form $AM = B_2 \cdot \text{coverage}$ were better than those of the form $AM = B_1 \cdot \text{size}$, in the sense of having a higher adjusted R^2 . However, we found that models of the form $AM = B_1 \cdot \text{size} + B_2 \cdot \text{coverage}$ were better than models of either of the previous two forms. This indicates that information about size alone or coverage alone does not yield as good a prediction of effectiveness as information about both size and coverage. This again suggests that both size and coverage influence effectiveness.

However, among the models that we studied, we found that models of the form $AM = B_1 \cdot \log(\text{size}) + B_2 \cdot \text{coverage}$ were always close to the best and usually were the best, in

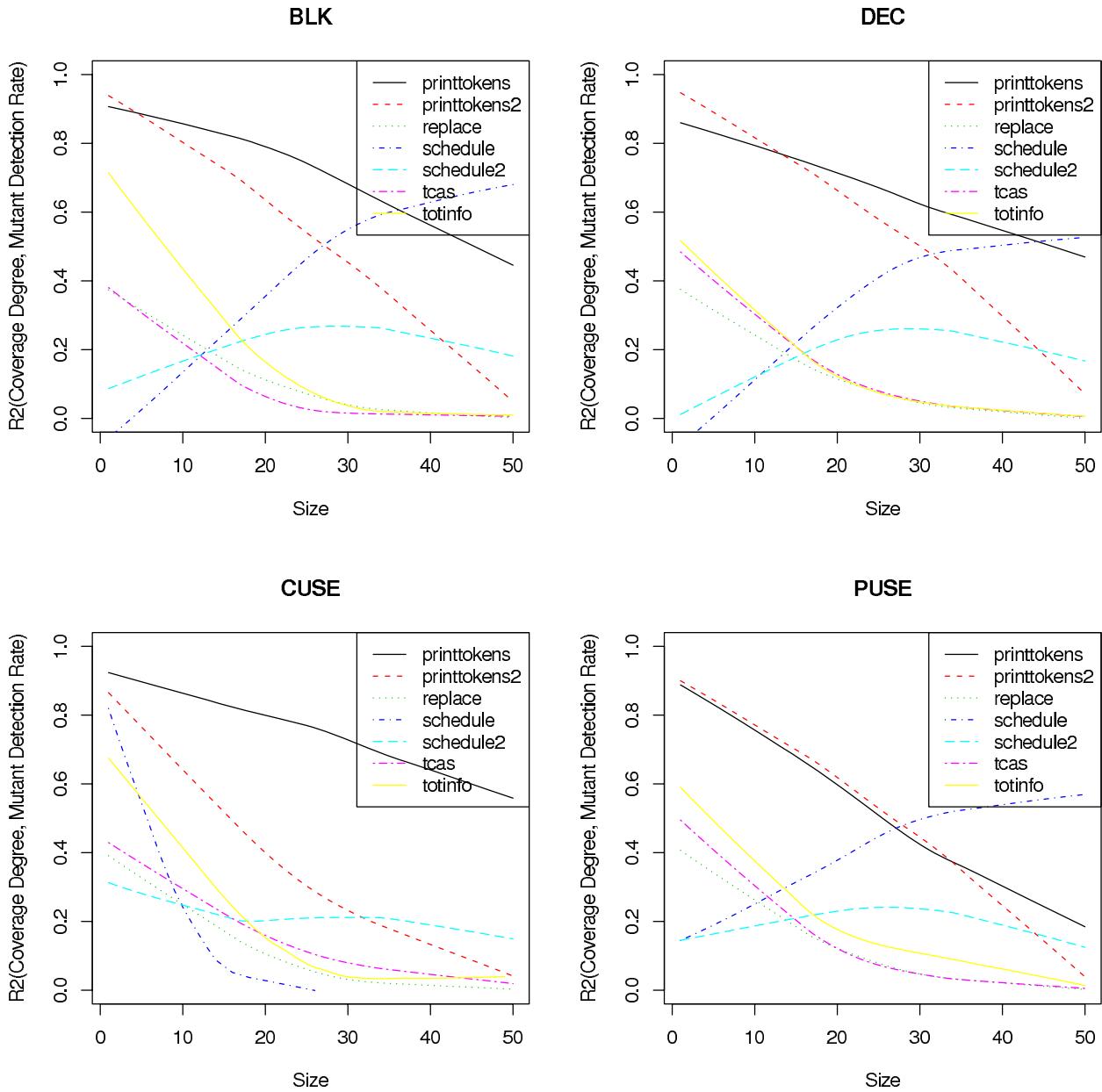


Figure 5: Correlation between coverage and mutant detection rate (Y-axis) varying with test suite size (X-axis).

Subject	Crit	Coefficient B1	B2	Adj. R^2	MSE
printtok	BLK	0.030	0.091	0.999	0.000
	DEC	2.433e-03	1.163e-01	0.999	0.000
	CUSE	5.465e-02	7.347e-02	0.999	0.000
	PUSE	2.671e-02	9.470e-02	0.998	0.001
printtok2	BLK	5.560e-02	6.338e-02	0.998	0.000
	DEC	3.195e-02	8.591e-02	0.998	0.000
	CUSE	4.914e-02	7.273e-02	0.998	0.001
	PUSE	1.934e-02	9.790e-02	0.998	0.000
replace	BLK	0.078	0.062	0.971	0.012
	DEC	0.048	0.087	0.971	0.011
	CUSE	0.075	0.067	0.970	0.012
	PUSE	0.047	0.087	0.972	0.011
schedule	BLK	4.650e-02	4.466e-02	0.998	0.001
	DEC	2.946e-02	5.915e-02	0.998	0.001
	CUSE	4.832e-02	4.841e-02	0.998	0.001
	PUSE	3.035e-02	5.319e-02	0.998	0.001
schedule2	BLK	4.288e-02	4.140e-02	0.997	0.002
	DEC	2.099e-02	6.170e-02	0.996	0.002
	CUSE	4.471e-02	3.969e-02	0.997	0.002
	PUSE	0.031	0.049	0.997	0.002
tcas	BLK	0.132	0.049	0.959	0.014
	DEC	0.072	0.099	0.960	0.014
	CUSE	0.125	0.054	0.960	0.014
	PUSE	0.081	0.091	0.960	0.014
totinfo	BLK	7.245e-02	7.430e-02	0.995	0.004
	DEC	6.030e-02	8.004e-02	0.994	0.004
	CUSE	0.087	0.071	0.994	0.004
	PUSE	6.807e-02	7.845e-02	0.994	0.004

Table 3: A summary of the linear models $AM = B1 \cdot \log(SIZE) + B2 \cdot CovDeg$. The “Crit” column gives the coverage criterion used (block, decision, C-use or P-use). The “MSE” column gives the mean squared error of the data points compared to the value predicted by the model.

the sense of having a higher adjusted R^2 than models based on raw size and coverage, on $\log(\text{size})$ alone, or on any other combination of factors.

Table 3 shows data concerning the $\log(\text{size}) + \text{coverage}$ models for the seven subject programs and four coverage measures, including the coefficients of the models and the adjusted R^2 values. Figure 6 shows graphs of actual (Y-axis) vs. predicted (X-axis) values for these $\log(\text{size}) + \text{coverage}$ models. Each graph also shows the $X=Y$ line and the LOWESS curve fitted to the data.

5. STUDIES ON OTHER PROGRAMS

In order to cross-check the results that we found with the Siemens programs, we studied two other programs with different characteristics to the Siemens programs: **gzip** and **concordance**. The results that we obtained confirmed that log-size-coverage models are a good fit.

Here we describe the subject programs that we used, how we collected the data, and the procedure and results of our statistical analysis of that data.

5.1 Subject Programs

5.1.1 gzip

The subject program **gzip**, also accessed through the Subject Infrastructure Repository at the University of Nebraska-Lincoln, is version 5 available at that repository of the standard GNU project file-compression utility. It contains 5680 net lines of code, and it comes with 214 test cases. We could not use two of the test cases because they did not work properly on our experimental platform, so we were left with 212 test cases.

5.1.2 concordance

The subject program **concordance** is a utility for making concordances (word indices) of documents. This paper introduces it as a subject program for the first time. The original program was written in C++ by Ralph L. Meyer, and made available under the GNU General Public Licence on various open-source websites. The source file consists of 966 net lines of C++ code, comprising five classes and 39 function definitions.

Over the course of several years, the second author and several classes of fourth-year students of a testing course at the University of Western Ontario identified 13 separate faults in the program. The source code was concatenated into one source file, and the faults were eliminated in such a way that they could be independently re-introduced to produce 13 faulty versions.

Six students in the testing course class of 2005 donated their test suites for **concordance**. These test suites were constructed according to guidelines for black-box testing (category-partition, boundary-value, extreme-value, syntax testing) and white-box testing (all test suites achieve maximum feasible decision/condition coverage as measured by BullseyeCoverage [3]). These test suites were put together to form a test pool with a total of 372 test cases.

5.2 Data Collection

In order to measure the effectiveness of test suites for **concordance**, we had previously generated 867 non-equivalent mutants automatically using the mutant generator used in

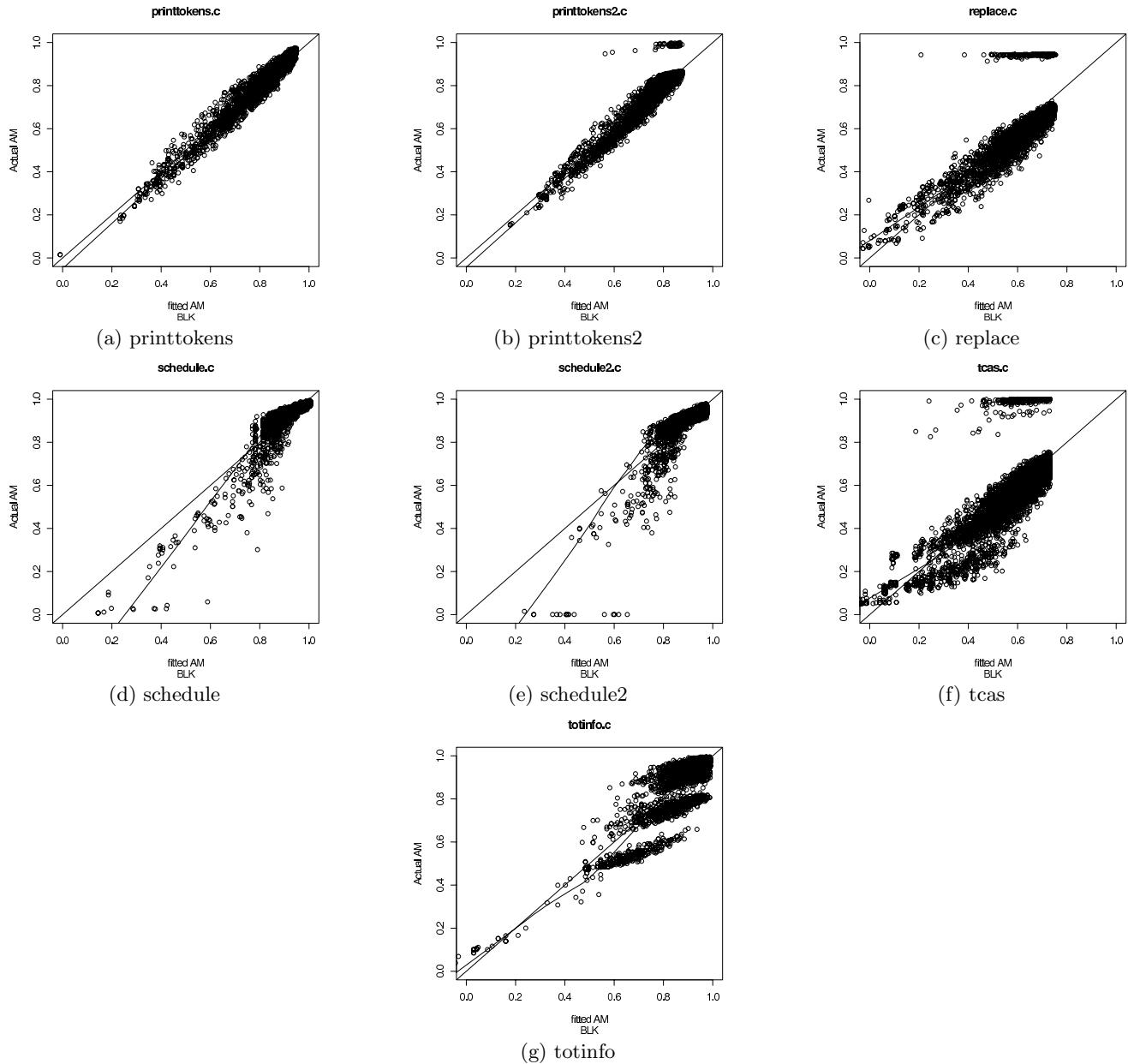


Figure 6: Predicted vs. actual AM for block coverage criterion using the model $AM \cong B1 \cdot \log(size) + B2 \cdot coverage$.

Model of AM or AF	Adjusted R^2		
	gzip AM	conc AM	conc AF
size	0.4796	0.8254	0.9127
coverage	0.8103	0.9973	0.9010
size + coverage	0.8259	0.9986	0.9579
log(size)	0.5563	0.9663	0.9628
log(size) + coverage	0.9905	0.9988	0.9643

Table 4: Goodness of fit of models of effectiveness. Effectiveness is measured by AM or AF; goodness of fit is measured by adjusted R^2 .

an earlier study [1]. Following the standard method for approximating non-equivalence, a mutant was judged as non-equivalent when no test case in the test pool was able to generate mutant output that was different from the gold version output. We ran all mutants on all test cases, and recorded which test cases killed which mutants.

In order to measure the effectiveness of test suites for gzip, we generated mutants automatically using Proteum [4]. Siami Namin et al. [14] had previously identified a set of 28 Proteum mutation operators that formed a sufficient set, i.e. a set of mutants such that the behaviour of subject programs on the sufficient set predicted their behaviour on all mutants. Proteum generated 38621 mutants of gzip using only these sufficient-set operators. Because each run of gzip took a relatively long time, it was not feasible to use more than a small fraction of these mutants. We therefore randomly selected 1% of the mutants from each operator, some of which did not compile; our final total was 317 mutants. We ran all mutants on all test cases, and recorded which test cases killed which mutants.

As with the Siemens programs, we generated 100 test suites of each size from 1 test case to 50 test cases, for each subject program. Each test suite was generated by randomly choosing each test case from the test pool, with a uniform distribution. We computed the coverage achieved by each test suite as the percentage of lines of code executed by the test suite, measured by the GNU coverage utility gcov. As a measure of effectiveness, we computed AF for **concordance**, i.e. the fraction of the 13 real faults detected by each test suite. We also computed AM for both programs, i.e. the fraction of the 317 mutants for **gzip** (resp. 867 mutants for **concordance**) detected by each test suite.

The immediate practical reason for the heterogeneity of the two subject programs and mutant generation was that the two authors were working in parallel on the two subject programs at different times and sometimes at different institutions. However, the heterogeneity strengthens our conclusions, since the results were similar for the two programs.

5.3 Data Analysis

We performed linear regressions for the two subject programs in a similar manner as we had for the seven Siemens programs. We modeled AM for both programs and AF for **concordance** with five models: size only, coverage only, size and coverage, log(size), and log(size) and coverage. We then measured the adjusted R^2 values of the models, as an indication of how well the models fitted the data.

Table 4 shows the results. Coverage alone was better than size alone at predicting AM for both programs, but size alone

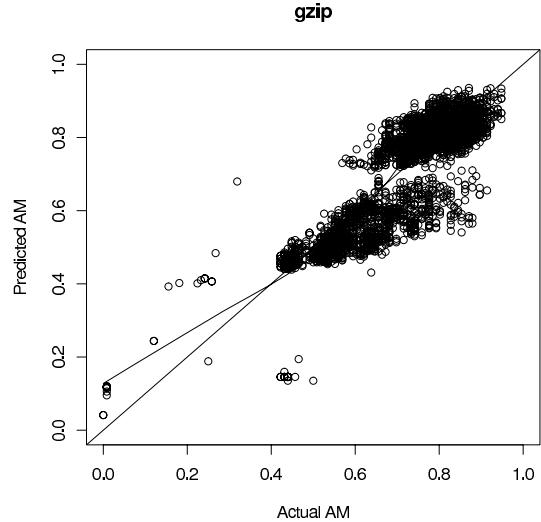


Figure 7: Actual AM of test suites for gzip (X-axis), and the value predicted for AM by the log(size) + coverage model (Y-axis).

was slightly better than coverage alone at predicting AF for **concordance**. Nevertheless, as with the Siemens programs, taking account of both size and coverage yielded a more accurate prediction of effectiveness than taking account of only one of them. Moreover, as with the Siemens programs, the log(size) + coverage models were the best, outperforming the models based on log(size) alone.

In order to see whether there was a statistically significant difference between the five models in each column of Table 4, we performed t tests with Bonferroni correction between each of the 10 pairs of models. The t tests indicated high statistical significance for the differences that we observed.

Figures 7, 8, and 9 plot the actual value of AM or AF (X-axis) against the value predicted by the corresponding log(size) + coverage model. Note that since only 13 faulty versions of **concordance** exist, the data points fall into a limited number of vertical bands (no test suite that we generated ever detected more than 12 of the faults). This is also the likely explanation of why the prediction for AF was not as good as for AM.

6. DISCUSSION

Although we found that the correlation between coverage and effectiveness is low for some test suite sizes and some programs, the correlation was sometimes high and seldom close to zero. This is probably why taking account of coverage data yielded a more accurate prediction of effectiveness over the entire range of test suite sizes than ignoring it.

Several analyses confirmed that there was no straightforward linear relationship between size, coverage and effectiveness. However, the accuracy of the log(size) + coverage models of effectiveness, and the consistency with which these models yielded accurate predictions, indicate that this relationship deserves further study.

Our working hypothesis as to why the log(size) + coverage models of effectiveness are accurate is as follows. Some faults are harder to find than others, so a given test case is more likely to find some faults than others. When a given test case

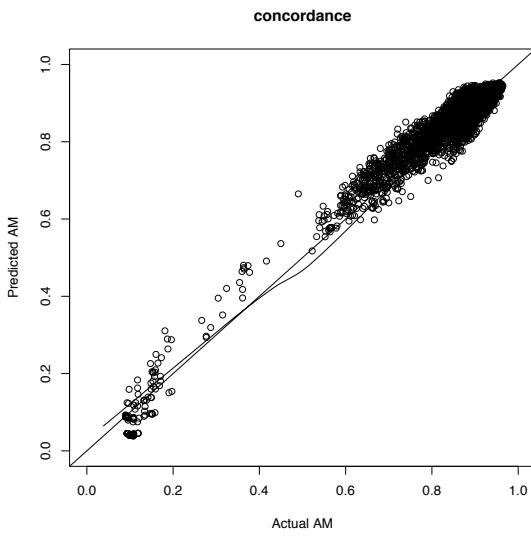


Figure 8: Actual AM of test suites for concordance (X-axis), and the value predicted for AM by the log(size) + coverage model (Y-axis).

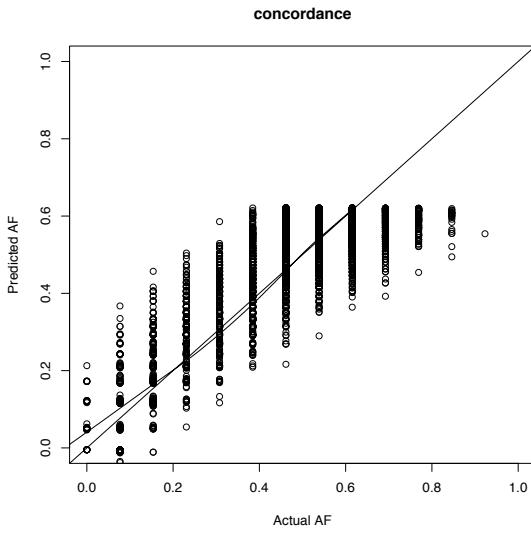


Figure 9: Actual AF of test suites for concordance (X-axis), and the value predicted for AF by the log(size) + coverage model (Y-axis).

C is added to a test suite S, it will increase the effectiveness of S only if it finds another fault. However, the faults that C is more likely to find are also the faults that S is more likely to have found already. So the bigger S is, the less likely it is (all other things being equal) that C will increase its effectiveness. Thus the gain in effectiveness is less and less for each new test case added. This helps explain the $\log(\text{size})$ component of the $\log(\text{size}) + \text{coverage}$ models.

However, faults tend to be associated with particular lines of code, decisions, or dataflow patterns. A test case that executes those program elements associated with the fault is much more likely to force a failure than one that does not. Therefore, regardless of the size of a test suite S, a fault is more likely to be exposed by a test case C that covers new program elements than one that does not. Thus if C achieves new coverage, it tends to increase the effectiveness of S regardless of the current size of S. This helps explain the coverage component of the $\log(\text{size}) + \text{coverage}$ models.

This work holds two main implications for software testers. First, achieving a high level of a strong coverage criteria leads directly to higher effectiveness of a test suite – not just indirectly by making the test suite bigger. Second, because of the $\log(\text{size}) + \text{coverage}$ relationship, when a test suite is being built, achieving higher coverage becomes more and more important relative to size as size grows. This implication reinforces longstanding advice by industry test professionals like Marick [10], who recommend testing first without directly considering coverage, and then looking at missing coverage in order to determine what classes of test cases could improve the test suite.

7. THREATS TO VALIDITY

Threats to internal validity include the possibility of corrupted data or incorrect procedures. The authors mitigated this threat by checking each other's work; we also performed various visualizations and complementary statistical analyses, one purpose of which was to check for anomalous results.

The constructs AM and AF are not the only possible measures of effectiveness. Measuring the effectiveness of a test suite at detecting more valuable errors (as defined, for instance, by an operational profile) may lead to different results. Similarly, the measurements of coverage that we used may not reflect all uses of coverage, such as coverage of sections of code deemed to be more important. We mitigated these threats to construct validity by using more than one measure of coverage and effectiveness.

The most serious threats are to external validity. We used programs in two different languages (C and C++), but only one of the programs was in C++, and the two languages are similar. Our results might not extend to other languages. Seven of our subject programs were small, with a maximum of 513 NLOC; only two were bigger, at 966 and 5680 NLOC. The results might therefore not extend to larger programs.

Finally, we worked with only randomly-selected test suites as a way of controlling for test suite construction methods, in order to make the most general conclusions possible about the influence of coverage on effectiveness. However, if we had restricted our attention to particular algorithms for building test suites, our results may well have been different, and while less general, may have applied more directly to real-world methods of test suite construction.

8. CONCLUSIONS AND FUTURE WORK

In order to determine whether coverage measures really are useful as adequacy criteria, we performed controlled experiments aimed at identifying the separate influences of test suite size and test suite coverage on test suite effectiveness.

Our results support the claim that both size and coverage independently influence test suite effectiveness, but not in a linear relationship. We identified a class of models (the log(size) + coverage models) that yield accurate predictions of test suite effectiveness on our data.

The program (*Concordance*) that we developed as a new experimental subject behaved similarly to the other programs. The program and its associated artifacts can be obtained by writing to the second author.

We would like to run similar experiments on programs from a broader range of programming languages, sizes and problem domains, in order to study how broadly the log(size) + coverage relationship with effectiveness applies. The major hurdle here will be the measurement of effectiveness, since mutant generators are often programming language-specific, and running every test case on every mutant of a large program can quickly lead to infeasibility. Parallel processing of many test cases at once may help here.

It would also be of interest to generate models of effectiveness of test suites from one set of programs, and see how accurately they predict the effectiveness of test suites on other programs. Using smaller programs as the training set and larger programs as the test set may avoid the high processing requirements imposed by larger programs.

9. ACKNOWLEDGMENTS

Thanks to Mats Heimdahl and Gregg Rothermel for helpful comments and suggestions, and to José Maldonado for the use of the Proteum mutant generator. Thanks also to the anonymous referees for valuable observations. Thanks to Rashin Arab, Tamilselvi Dhandapani, Desmond Lang, Florian Niedermann, Yosua Setyobudhi, and Si Shen for contributing their test suites to the development of *concordance*. Special thanks to Tamilselvi Dhandapani and Shaimaa Ali for their work in data collection for *concordance*. This research was supported by NSERC Canada.

10. REFERENCES

- [1] J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, St. Louis, Missouri, May 2005. 402–411.
- [2] J. H. Andrews, L. C. Briand, Y. Labiche, and A. Siami Namin. Using mutation analysis for assessing and comparing testing coverage criteria. *IEEE Transactions on Software Engineering*, 32(8):608–624, August 2006.
- [3] Bullseye Technologies. BullseyeCoverage website. www.bullseye.com, accessed January 2009, 2009.
- [4] M. E. Delamaro and J. C. Maldonado. Proteum – a tool for the assessment of test adequacy for C programs. In *Proceedings of the Conference on Performability in Computing Systems (PCS 96)*, pages 79–95, New Brunswick, NJ, July 1996.
- [5] P. G. Frankl and O. Iakounenko. Further empirical studies of test effectiveness. In *Proceedings of the 6th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 153–162, Lake Buena Vista, FL, USA, 1998.
- [6] P. G. Frankl and S. N. Weiss. An experimental comparison of the effectiveness of branch testing and data flow testing. *IEEE Transactions on Software Engineering*, 19(8):774–787, August 1993.
- [7] J. P. Guilford. *Fundamental Statistics in Psychology and Education*. McGraw-Hill, New York, 1956.
- [8] J. Horgan and S. London. ATAC: A data flow coverage testing tool for C. In *Proceedings of the Symposium on Assessment of Quality Software Development Tools*, pages 2 – 10, May 1992.
- [9] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proceedings of the 16th International Conference on Software Engineering (ICSE 1994)*, pages 191–200, Sorrento, Italy, May 1994.
- [10] B. Marick. How to misuse code coverage. URL www.example.com/testing-com/writings/coverage.pdf. 13pp., 1997.
- [11] A. J. Offutt and R. Untch. Mutation 2000: Uniting the orthogonal. In *Mutation 2000: Mutation Testing in the Twentieth and the Twenty First Centuries*, pages 45–55, San Jose, CA, October 2000.
- [12] G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In *Proceedings of the International Conference on Software Maintenance (ICSM '98)*, pages 34–43, Washington, DC, USA, November 1998.
- [13] G. Rothermel, M. J. Harrold, J. von Ronne, and C. Hong. Empirical studies of test-suite reduction. *Software Testing, Verification and Reliability*, 12:219–249, 2002.
- [14] A. Siami Namin, J. H. Andrews, and D. J. Murdoch. Sufficient mutation operators for measuring test effectiveness. In *Proceedings of the 30th International Conference on Software Engineering (ICSE 2008)*, pages 351–360, Leipzig, Germany, May 2008.
- [15] W. N. Venables, D. M. Smith, and The R Development Core Team. An introduction to R. Technical report, R Development Core Team, June 2006.